2019. 5. 15.

Assignment # 3-3

금요일 - 이성원 교수님

2015722087 컴퓨터정보공학부

김민철

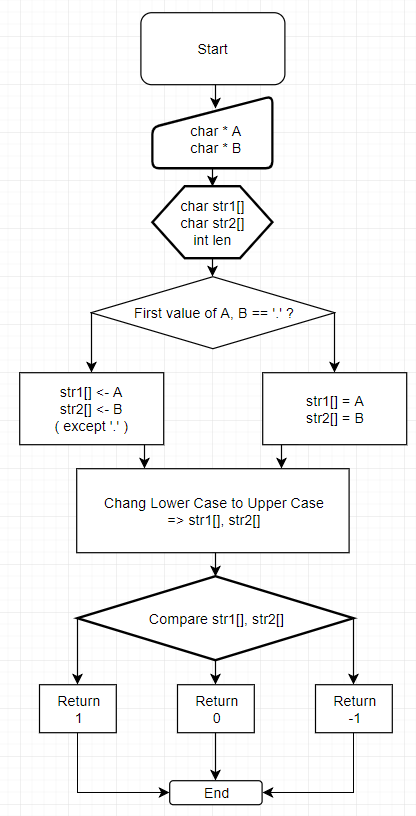
advanced web-server구현

**Introduction**

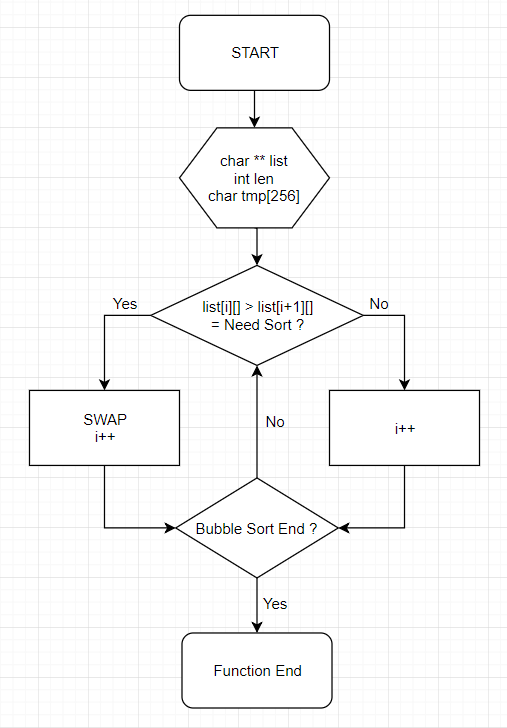
이번 과제에서는 3-2 과제에서 구현하였던 basic web-server 의 기능에서 다중 접속 지원 기능과 접근 제어 기능을 추가하는 과제이다. 다중 접속 지원은 다수의 클라이언트가 동시에 서버에 접근할 수 있게 하는 기능이고, 접근 제어 기능은 미리 허용된 IP의 사용자만이 서버에 접속할 수 있도록 하는 것이다. 이때 허용된 IP의 주소는 파일 (accessible.usr) 에 저장하고, 허용된 IP주소가 아닐 경우 에러 메세지 페이지를 출력한다.

**Flow Chart**

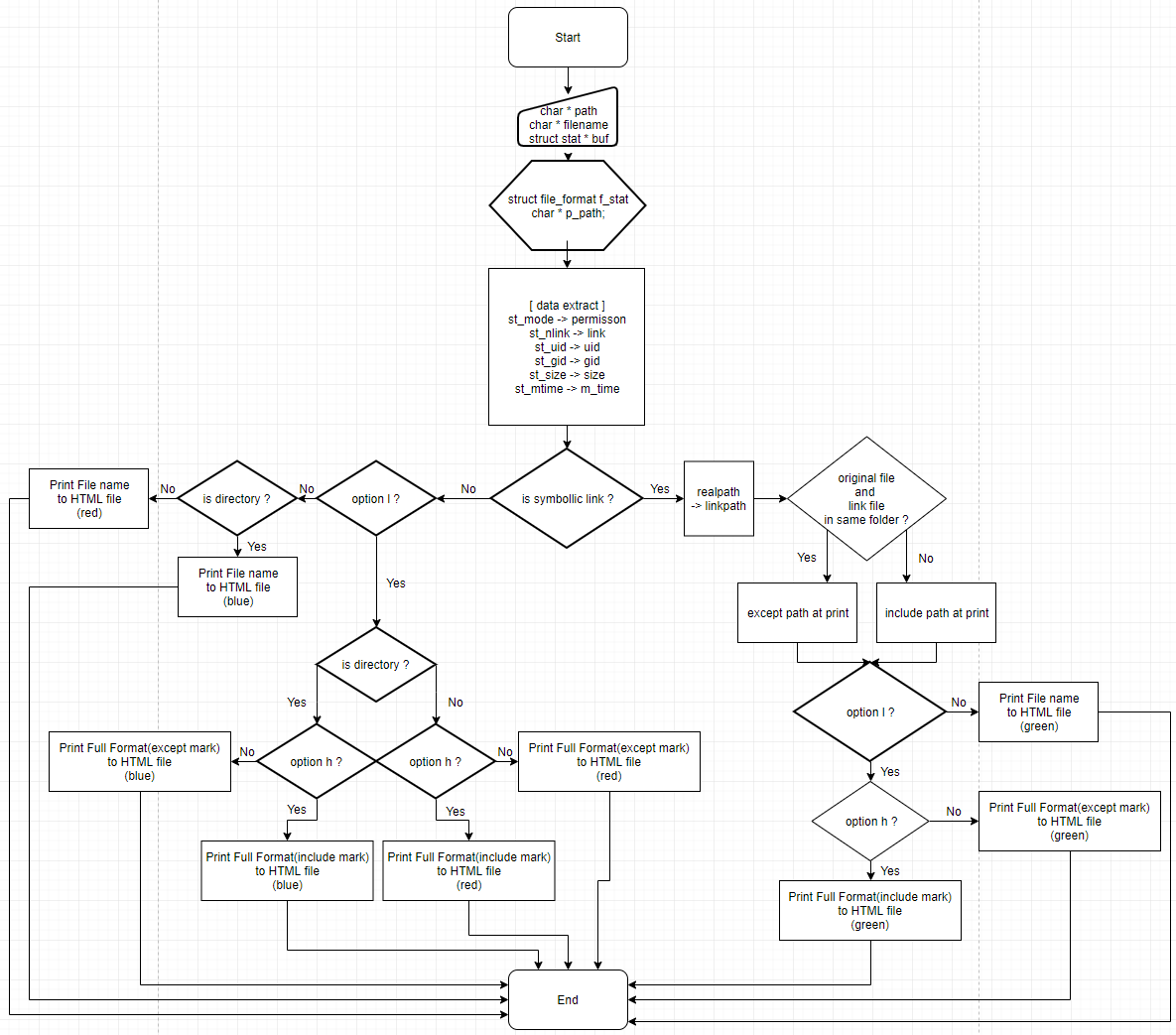
-int strcmp\_i



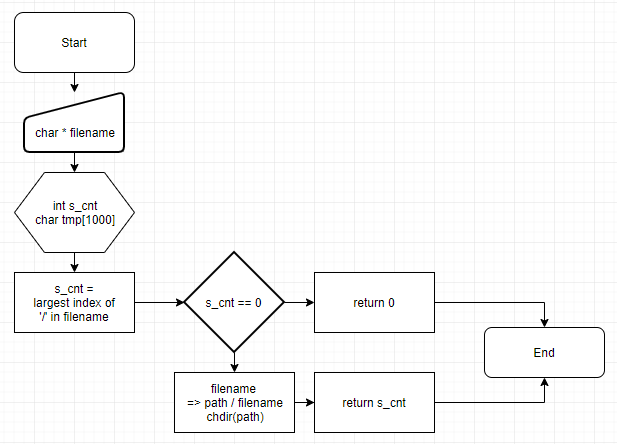
-void sort\_list



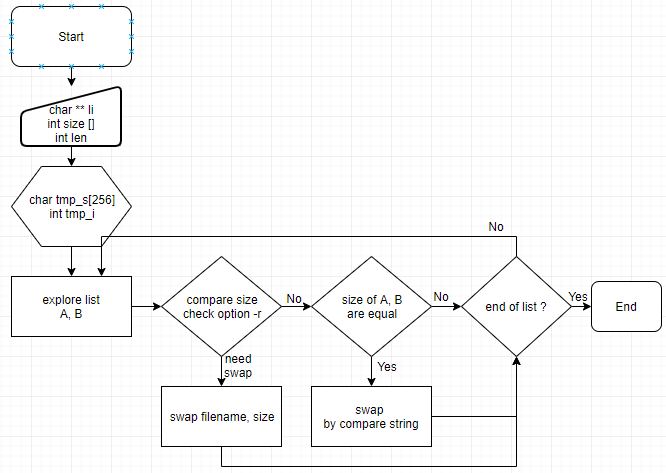
-void view\_advanced\_list



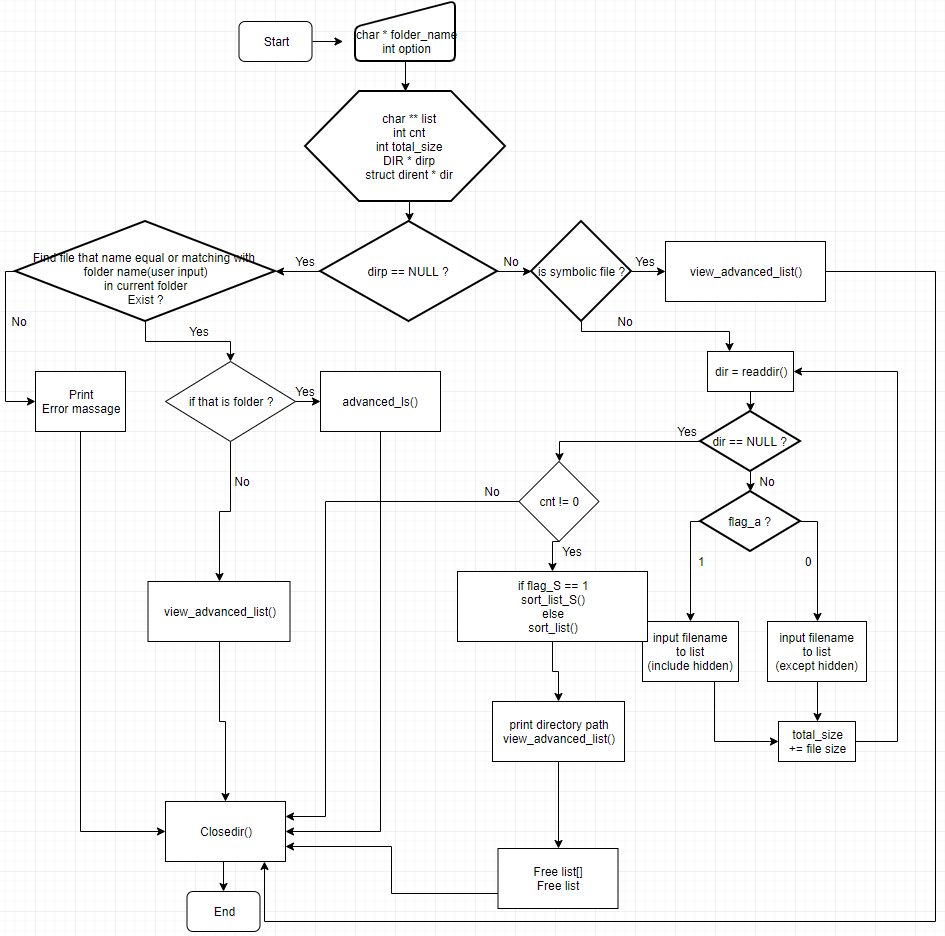
-int check\_real\_path



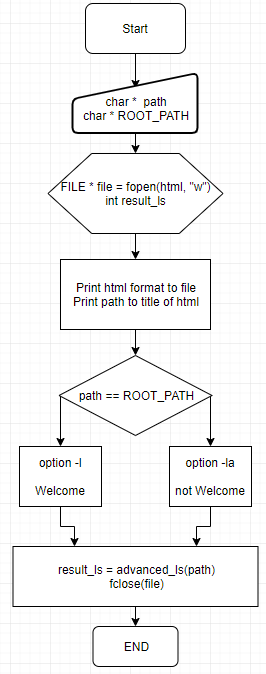
-void sort\_list\_S



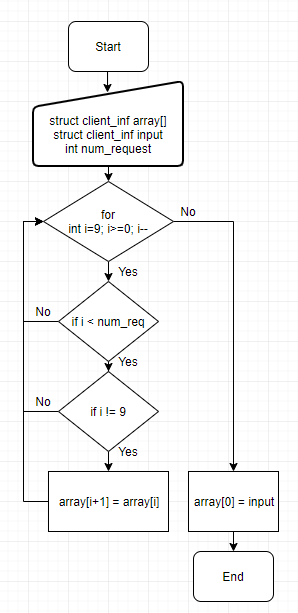
-int advanced\_ls



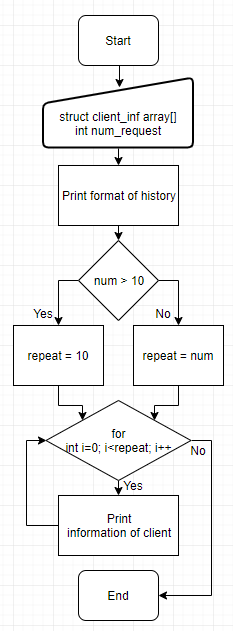
- int html\_ls



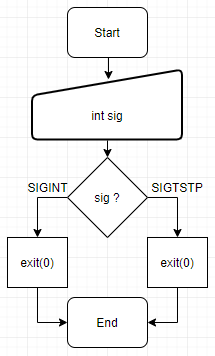
- add\_client



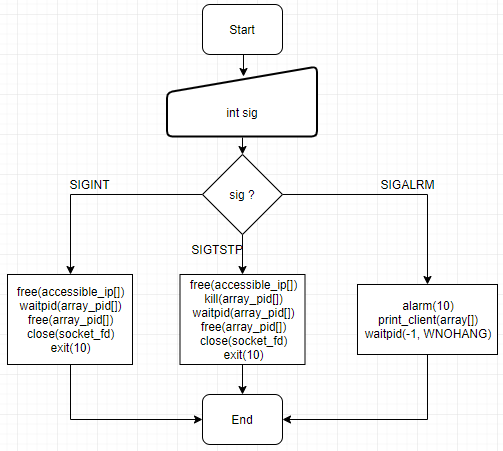
- print\_client



- signal\_handler\_c



- signal\_handler\_p



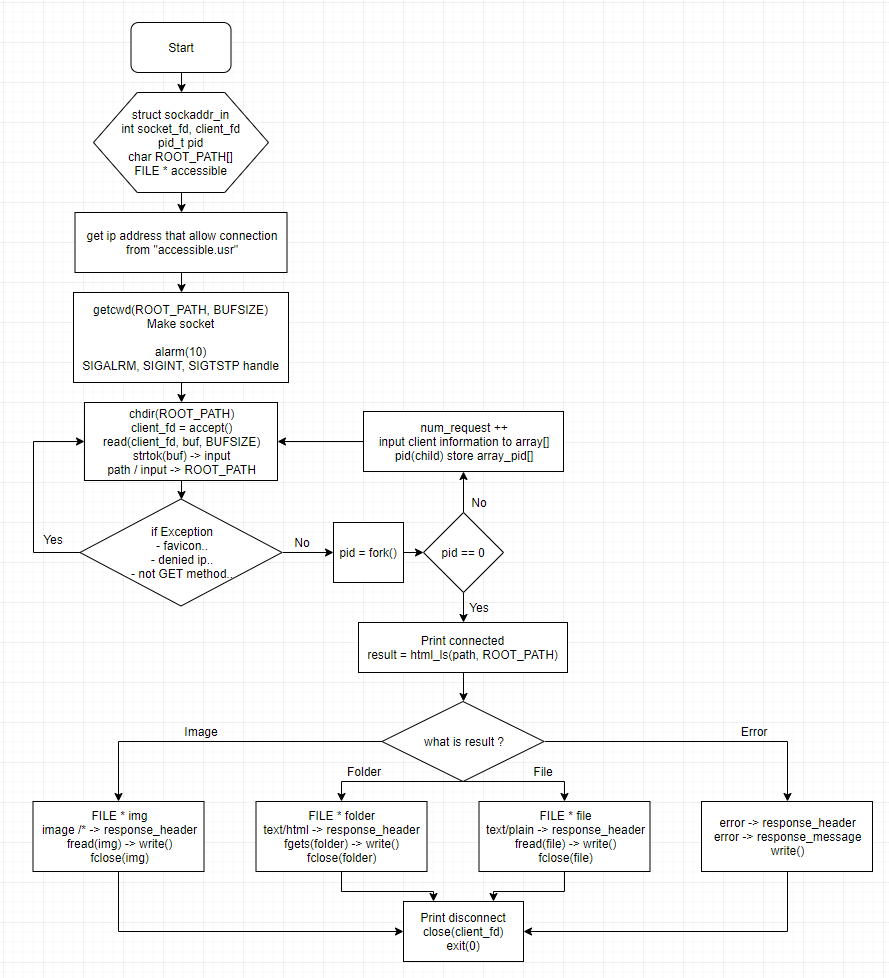
- int main

folder

file

ERROR

image



**Pseudo code**

int check\_real\_path(char \* filename)

{

for (int i = 0; i < strlen(filename); i++) // check directory.

{

입력된 파일 이름에 '/' 가 있을 경우 해당 index => s\_cnt

}

if 파일 이름에 '/' 가 있다면 -> real path

{

devide < path > / < file >

<path> 로 작업중인 디렉토리 변경

s\_cnt 값 반환

}

else real path 가 아니라면

0 반환

}

void swap(char \* str1, char \* str2)

{

str1과 str2 swap

}

int strcmp\_i ( char \* 비교할 첫 번째 문자열 A, char \* 비교할 두 번째 문자열 B )

{

<예외 처리> : r 옵션이면 거꾸로.

A랑 B가 “.” , “..” 이면 “..” 이 뒤로,

둘 중 하나만 “.” 이면 “.”이 앞으로

둘 중 하나만 “..” 이면 “..”이 앞으로

if( A 가 B 보다 길면 )

len = B의 길이;

else

len = A의 길이

for(len 만큼 반복)

{

if(소문자라면)

대문자로 바꾼다

}

If 옵션이 –r 이라면

if( A > B ) return = -1;

else if( B > A ) return = 1;

else return = 0;

else

if( A > B ) return = 1;

else if( B > A ) return = -1;

else return = 0;

}

void sort\_list(char \*\* li, int len)

{

for (저장된 파일이름 개수-1 만큼 반복)

{

if (더 이상 내용이 없다)

break;

for (저장된 파일이름 개수-1 만큼 반복)

{

if ( 앞에 저장된 파일이름 > 뒤에 저장된 파일 이름)

두 파일 이름을 swap

}

};

}

void sort\_list\_S(char \*\* li, int size[], int len)

{

// sort like bublle sort

for (int i = 0; i < len - 1; i++) // total cycle

{

if 리스트가 끝났다면

break;

for (int j = 0; j < len - 1; j++) // one cycle

{

if 파일 사이즈가 다르다면, 정렬 필요 -r 옵션에 따라서.

{

swap file name

swap file size

}

else if 파일 사이즈가 같다면, -> 문자열로 정렬

{

if 정렬이 필요하다면

{

swap file name

}

}

}

}

return;

}

void view\_advanced\_list(char \* 파일 경로, char \* 파일 이름, struct stat \* 파일 정보)

{

St\_mode에 저장되어 있는 파일 타입에 따라 permission[0] 결정

St\_mode에 저장되어 있는 user 권한에 따라 permission[1~3] 결정

St\_mode에 저장되어 있는 group 권한에 따라 permission[4~6] 결정

St\_mode에 저장되어 있는 other 권한에 따라 permission[7~9] 결정

* Permission = “- --- --- ---”

st\_nlink -> f\_stat->link

st\_uid -> f\_stat->uid

st\_gid -> f\_stat->gid

st\_size -> f\_stat->size

st\_mtime -> f\_stat->m\_time

if 옵션이 –h 라면

size = size / 1024.0, check\_h++(단위가 몇번 상승했는지 체크)

check\_h 값에 따라서 단위(K, M, G) 결정

// Full Format 출력

if 파일 type 이 symbolic link 라면

{

if l옵션이 아니라면

파일 이름만 HTML 파일로 출력 (green)

else Full Format 출력

If 옵션이 –h 라면, 사이즈 출력시 단위와 함께 출력

Symbolic link 파일과 가리키는 경로 파일이 같은 폴더에 있다면 -> 가리키는 파일명만 HTML 파일로 출력(green)

Symbolic link 파일과 가리키는 경로 파일이 다른 폴더에 있다면 -> 절대 경로 모두 HTML 파일로 출력(green)

}

else symbolic link 가 아니라면

{

if ㅣ옵션이 아니라면

directory일 경우 파일 이름만 HTML 파일로 출력 (blue)

아닐 경우 파일 이름만 HTML 파일로 출력 (red)

else l 옵션 이라면

if directory라면 (blue)

If 옵션이 –h 라면, FullFormat 출력시 단위도 HTML 파일로 출력

else FullFormat HTML 파일로 출력

else directory가 아니라면 (red)

If 옵션이 –h 라면, FullFormat 출력시 단위도 HTML 파일로 출력

else FullFormat HTML 파일로 출력

}

int advanced\_ls(char \* 입력한 폴더)

{

if 폴더가 아니라면

{

혹시 절대경로로 입력된 폴더인지 확인 -> 맞으면 해당 디렉토리로 변경

-> 아니면 현재 디렉토리

HTML 파일이면 return

if table이 만들어져 있지 않고, flag\_p가 -1이 아니라면,

옵션 l 에 따라 테이블 생성 -> flag\_table=0;

while 디렉토리 탐색

{

if 파일을 찾았다면

view\_advanced\_list()

If fnmatch 로 매칭이 된다면,

폴더면 다시 advanced\_ls()

파일이면 view\_advanced\_list()

}

If 출력한 파일이 없다면

에러 메시지 출력 -> 반환 -1

else 1 반환

}

else // is folder

{

p 플래그가 표시되어있다면 return 3 => 폴더

flag\_table == 0 이라면 테이블이 열려있으므로 close

flag\_table = 1 set -> 테이블이 안만들어져있다.

심볼릭 링크로 연결된 폴더인지 확인

-> 심볼릭 링크 폴더이고 옵션이 l 이면 심볼릭 링크만 Full Format 출력

아니면 탐색 시작

while 입력된 폴더 탐색

{

HTML 파일은 처리하지 않음.

if 파일이 없다면

반복문 탈출

if 히든 파일이고 옵션이 –l 이나 default 라면

{

List에 파일 이름 입력

Total size += 파일 size

}

else if 옵션이 –a 이나 –la 라면 모든 파일 입력

{

List에 파일 이름 입력

Total size += 파일 size

}

}

Directory path 출력

if 파일이 존재한다면

{

List 정렬.

Total size 출력

if l 옵션이라면 -> Full Format Table 생성

else -> 파일 이름 Table 생성

for 파일 개수 만큼 반복

view\_advanced\_list()

테이블 close

flag\_table=1

}

}

폴더 닫기

}

int html(char \* 찾을 경로, char \* 루트 경로)

{

origin\_wd에 루트 경로 복사

HTML 파일 기본 포맷 입력, 타이틀에 현재 디렉토리 입력.

HTML\_FILE 에 쓰기위한 fopen

title에 찾을 경로 입력

찾을 경로가 루트 경로이면 -l옵션하고 Welcome, 아니면 -la 옵션하고 Welcome 생략

result = advanced\_ls(path);

테이블 close

html 파일 포맷 close

open한 파일 close

result 반환.

}

- int main

{

"accessible.usr" 파일 오픈

접근 허용된 IP주소 읽어와서 accessible\_ip 배열에 저장

소켓 생성 -> 연결 준비

SIGALRM, SIGINT, SIGTSTP 시그널 처리, 10초뒤 SIGALRM 신호 발생

while(1)

{

ROOT PATH로 디렉토리 변경

클라이언트로 부터의 연결을 기다림

연결이 되면 예외처리 확인 (EXIT, favicon.ico)

접근 허용된 IP주소 확인(strcmp, fnmatch)

pid = fork() 포크 선언

자식 프로세스인 경우

SIGINT, SIGTSTP 신호 처리 선언

연결 상태 출력

루트 경로가 아닌 경우 루트 경로 / input -> path

루트 경로인 경우 루트 경로 -> path

result 에 html\_ls(path) 의 결과 출력 (폴더:3, 파일:1, 오류:-1)

if 이미지 파일인 경우

이미지 파일 open.

헤더 메세지 : image/\*

응답 메세지에 이미지 파일 내용 전송

else if 폴더인 경우

html\_ls의 결과인 html 파일 open

헤더 메세지 : text/html

응답 메세지에 html 파일 내용 전송

else if 파일인 경우

파일 open

헤더 메세지 : text/plain

응답 메세지에 파일 내용 전송

else

에러 예외처리 메세지 전송

연결 종료.

프로세스 종료

부모 프로세스인 경우

연결된 클라이언트 정보 array에 저장

자식의 pid 를 array\_pid 배열에 저장

}

접근 허용 ip 배열 동적할당 해제

wait 하지 않은 자식 프로세스 waitpid

소켓 종료

return 0

}

void add\_client(struct client\_inf in\_array[], struct client\_inf input, int num)

{

if index < num

if index != 9

array[i]에 있는 데이터를 array[i+1]로 복사.(이동)

새로운 데이터 0번 index에 입력

}

void print\_client(struct client\_inf in\_array[], int num)

{

양식 출력 -> request number 출력

if (num > 10)

repeat = 10

else

repeat = num

repeat 만큼 반복

array에 들어있는 연결 정보 출력

}

void signalHandler\_c(int sig)

{

if sig == SIGINT

exit

if sig == SIGTSTP

exit

}

void signalHandler\_p(int sig)

{

if sig == SIGINT

free accessible\_ip

waitpid (모든 자식)

close(소켓)

exit

if sig == SIGTSTP

free accessible\_ip

kill (모든 자식)

close(소켓)

exit

if sig == SIGALRM

10초뒤 알람 설정

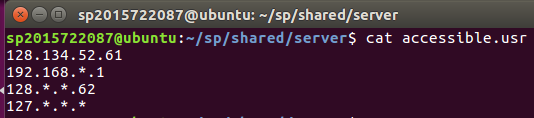
if num\_request > 0

print\_client();

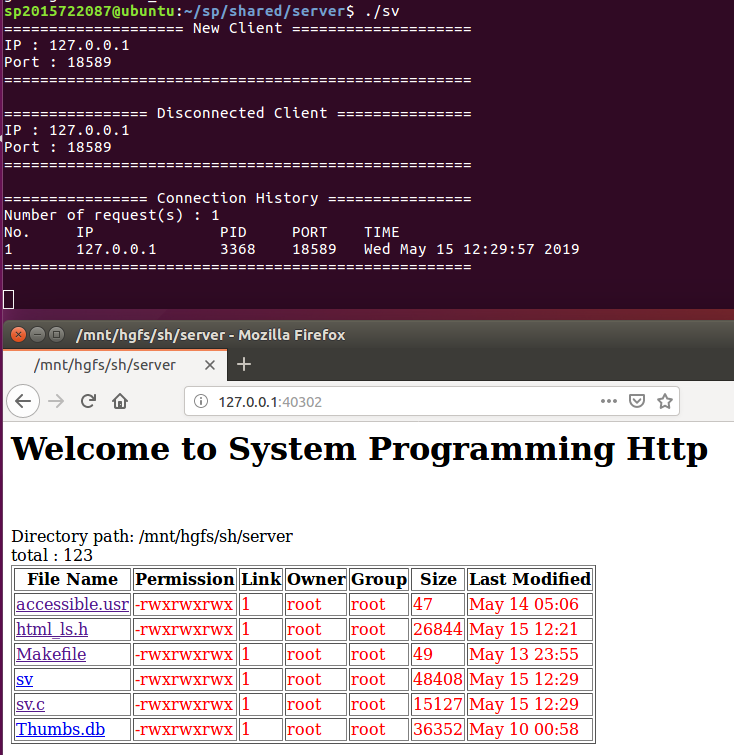
waitpid (모든 자식)

}

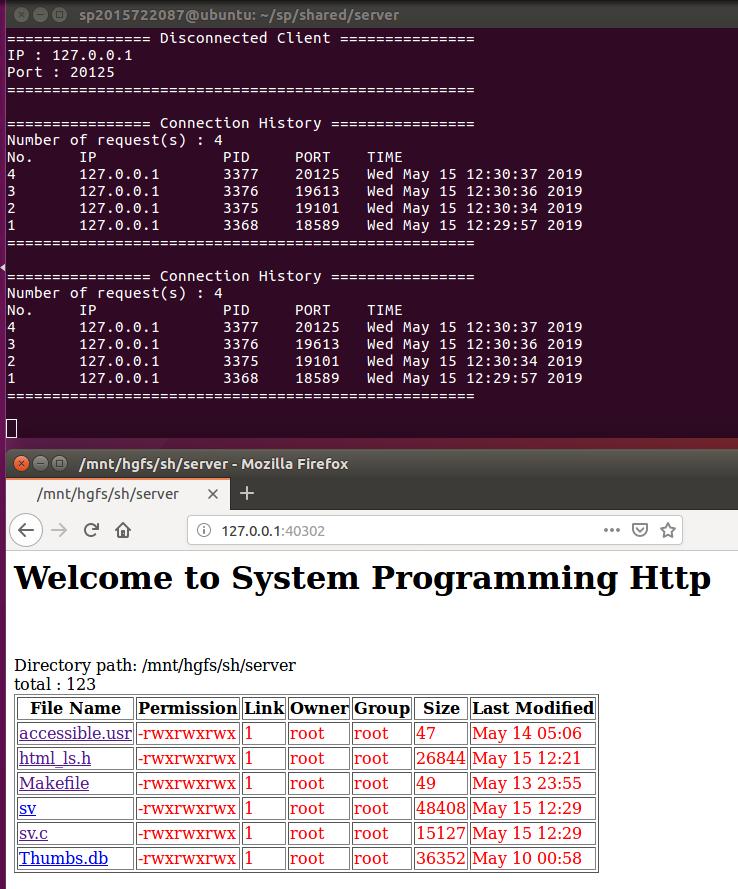
**Result**



- 현재 accessible.usr 파일의 내용이다. 127.0.0.1 은 접속이 가능하다.



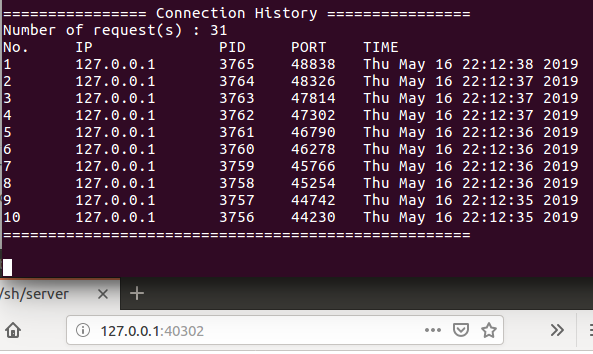
- 웹서버가 실행되고, 클라이언트로부터 연결 요청을 받아 연결 후 결과를 브라우저로 출력한 모습이다. 10초마다 Connection History를 출력하는 모습이다.



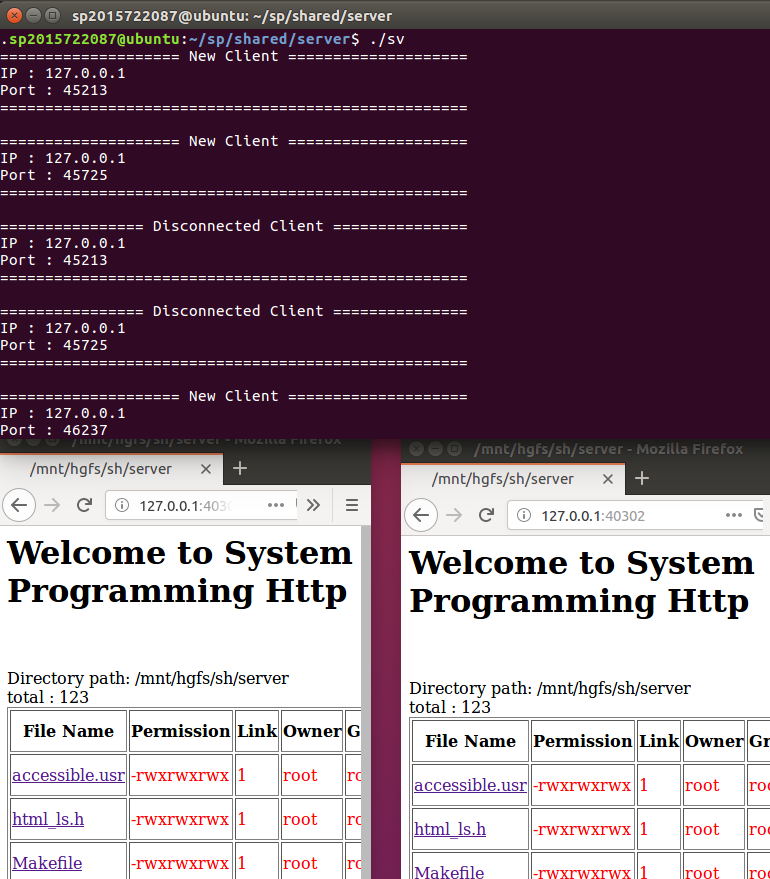
- 연결 요청이 여러개 일때 Connection History의 모습이다. Number of request는 총 서버 프로그램의 총 연결 횟수이다. 그 아래로 No. 는 몇번 째로 연결된 것인지 구분해주는 값이고, IP 는 클라이언트의 IP 주소, Port와 PID 는 클라이언트를 위한 포트 번호와 프로세스 아이디이고, Time은 서버와 클라이언트가 연결된 시간이다.

최신 시간 순서대로 정렬하였기 때문에 늦게 (최근에) 연결된 클라이언트의 정보(No가 높은 값)이 가장 먼저 출력되었다.

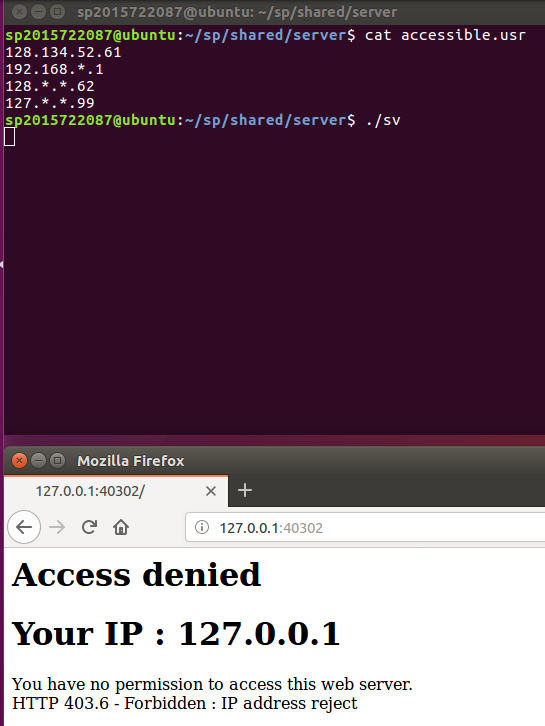
(No번호가 잘못 입력되었습니다 수정된 캡쳐화면 아래에 첨부하였습니다)



- 10개 이상의 연결이 기록이 남을때는 가장 최근 시간에 연결된 기록 10개까지만을 출력한다.



- 브라우저 두개를 열어 동시에 접속을 시도한 모습이다. New Client는 연결이 완료 되었을 때 나오는 출력이므로 두개의 New Client가 출력된 것으로 보아 두 연결 모두 진행되어 다중 접속이 가능함을 알 수 있다.



- accessible.usr 파일을 변경 한 후 접근이 거절된 모습이다. 127.0.0.1이 접속 허용이 안되어 있으므로 허가되지 않은 IP의 접속이라는 에러 메세지를 출력한다.

**Conclusion**

이번 과제는 지난번의 과제를 기반으로 기능을 추가하는 과제라 처음으로 강의 자료를 받아 보고 과제를 확인하고 나서 걱정이 되지는 않은 과제이다. 코드는 예시 코드와 설명이 주어져있어서 금방 작성하였지만, 내가 코드를 이해하고 작성하고 있는 것이 맞는지 모르겠다는 생각이 들어서 강의 자료를 보면서 공부를 하면서 코드를 작성하였다.

지난번 과제를 진행할때 작성했던 코드도 다시 보니 왜 적혀있는지, 왜 꼭 여기에 적혀있어야 되는지 헷갈리는 것이 있어 이번 3차 과제는 할만 할지 몰라도 퀴즈는 어려울 것 같다는 생각이 들어 이론적인 이해가 필요하다는 생각이 들었다.